42P18129

Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT APPLICATION

SHARING IDLED PROCESSOR EXECUTION RESOURCES

Inventor(s):

Koichi Yamada and Allen M. Kay

Prepared by:

Blakely, Sokoloff, Taylor & Zafman, LLP
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA 90025-1026
(503) 684-6200

Express Mail Label: EV325529242US

Sharing Idled Processor Execution Resources

Background

[01] In the high level view of a processor depicted in Fig. 1a, a processor may be conceptualized as being comprised of two components, the first implementing the architectural state of the processor, such as for example its registers and program counter, and the second composed of processor execution resources, such as, for example, a translation lookaside buffer (TLB).

[02] In one type of multiprocessing processor based system, as depicted in Fig. 1b, multiple physical processors are interconnected by a bus system, and each physical processor maintains a separate architectural state in hardware as well as a separate set of processor execution resources in hardware. In a thread scheduling scenario where each processor of such a system is scheduled to execute a different thread, an instance may arise when one of the processors in the system is idled because it is waiting on a slower device in the system, such as a disk drive, or because it is currently not scheduled to execute a thread. In this instance, the processor and all of its execution resources are also idled and unavailable to other processors of the system.

[03] In another type of processor based system such as that depicted in Fig. 1c, a hardware processor that maintains separate architectural states in the processor's hardware for a plurality of logical processors may, however, have a single processor core pipeline that is shared by the logical processors and a single set of processor execution resources, including the TLB, that is shared by the logical processors. Such a processor architecture is exemplified by the Intel® Xeon™ processor with Hyper Threading Technology, among others, and is well known in the art.

[04] In such a logical multiprocessing system, a thread scheduler may schedule a different thread to execute on each of the logical processors because each logical processor maintains its architectural state separately from all other logical processors.

When a logical processor is idled by an operating system thread scheduler or is waiting for data from a slow storage device, it may either execute an idle task, typically a tight loop, and periodically check for an interrupt; or it may suspend its activity and wait for a wake up signal of some type to resume execution of a thread.

[05] In contrast to a multiprocessing system where processor execution resources are physically separated, in this type of logical multiprocessing system, when one of the multiple logical processors in such a system is idled, dynamically allocated processor execution resources that are not being used by the idled logical processor may be available to other logical processors that are currently executing threads for the user or the system.

[06] Processor execution resources in a logical multiprocessing system may, however, be reserved for a logical processor. This may occur in different ways. For one example, a logical processor may lock a dynamically allocated processor execution resource such as a translation register (TR) from the TLB thus making it unavailable to other logical processors. In another instance, the logical processor may be statically allocated processor execution resources such as TCs and thus these statically allocated resources may be unavailable to other logical processors. These reserved resources typically continue to be unavailable to other logical processors even after the logical processor for which they are reserved is idled. Thus, TRs that are locked by a logical processor generally continue to be locked by the logical processor while it is idling; and statically allocated TCs allocated to the logical processor continue to be statically allocated to the logical processor while it is idling.

Brief Description of the Drawings

Figure 1 Depicts high level views of different types of processor architectures.

Figure 2 is a flowchart of processing in one embodiment.

Figure 3 depicts a processor based system in one embodiment.

Detailed Description

[07] In one embodiment processing occurs as depicted in the high level flowchart in Fig. 2. In the figure, two logical processors, Processor 1, 200, and Processor 2, 205, are executing threads scheduled by an operating system that includes a thread scheduler 210. At 215, Processor 1 is switched out from an executing thread due to, for instance, termination of the thread or a page fault, and returns to the thread scheduler. If no more tasks are scheduled for this logical processor, 220, the processor executes an idling sequence, 225-230. First, the logical processor gives up any reserved processor execution resources held by the logical processor 225, releasing them to the common pool 260. Thus for example, Processor 1 may return a Translation Cache entry or Translation Cache Register to the general pool of registers in the Translation Lookaside Buffer.

[08] In different embodiments, the processing in step 225 may differ. In some embodiments, the exclusively held resource released may be a dynamically allocated resource and have previously been locked by Processor 1. In such an embodiment, in step 225, the logical processor unlocks the resource and thereby makes it available to other logical processors. In another embodiment, the exclusively held resource may have been previously statically allocated to Processor 1. In such embodiments, in step 225, the statically allocated resource is deallocated and is returned to the pool of dynamically allocated resources 260.

[09] After Processor 1 enters an idled state, such as a state of suspension 230 in this embodiment, it may be requested for execution of a new or resumed thread by a wake up signal such as an interrupt 235. In other embodiments the processor may enter an idle task loop instead of the suspension depicted at 230 and periodically check for interrupts.

[10] Following the wake up signal, the logical processor then re-acquires the exclusively reserved resources by either locking or statically allocating them to itself as necessary, 240. The logical processor then switches to an incoming thread and continues execution of that thread, 245.

[11] The resources freed by Processor 1 before suspension or idling at 225 become available to another logical processor such as Processor 2, 205, executing a thread such as the depicted user thread 250. These resources may then be dynamically allocated to the logical processor as necessary from the pool of shared processor execution resources during the execution of the thread, 255.

[12] Fig. 3 depicts a processor based system in one embodiment where the logical processors are implemented as part of a processor 300. Programs that execute on the logical processors are stored in memory 340 connectively coupled to the processor by bus system 320. The memory may include a non-volatile memory section storing firmware that includes a thread scheduler performing processing substantially as described above.

[13] Many other embodiments are possible. For instance, while the above description limits itself to logical processors, similar processing is applicable to physically separate multiprocessors that share any common execution resources. In such embodiments, a hybrid version of logical and physical multiprocessing is implemented where separate architectural states and some execution resources are separated in hardware, but other execution resources are shared in hardware and may be released using processing similar to that depicted in Fig. 2. In some embodiments, the thread scheduler referenced above may form a component of firmware resident in non-volatile memory as depicted in Fig. 3, while in others it may be a portion of operating system software stored on disk media accessible to the processor. In some embodiments, the actions taken to release and reserve processor execution resources may be directly implemented in hardware and ancillary to the processor's instruction

execution system, while in other embodiments they may be actions taken by the processor as part of the execution of one or more instructions. In some embodiments the shared execution resources may include special purpose registers unrelated to the TLB. Embodiments are not limited to two processors, three or more processors may share execution resources and perform processing analogous to the processing described above.

[14] Embodiments in accordance with the claimed subject matter may be provided as a computer program product that may include a machine-readable medium having stored thereon data which when accessed by a machine may cause the machine to perform a process according to the claimed subject matter. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, DVD-ROM disks, DVD-RAM disks, DVD-RW disks, DVD+RW disks, CD-R disks, CD-RW disks, CD-ROM disks, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, magnet or optical cards, flash memory, or other type of media / machine-readable medium suitable for storing electronic instructions. Moreover, embodiments may also be downloaded as a computer program product, wherein the program may be transferred from a remote computer to a requesting computer by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

[15] Many of the methods are described in their most basic form but steps can be added to or deleted from any of the methods and information can be added or subtracted from any of the described messages without departing from the basic scope of the claimed subject matter. It will be apparent to those skilled in the art that many further modifications and adaptations can be made. The particular embodiments are not provided to limit the invention but to illustrate it. The scope of the claimed subject matter is not to be determined by the specific examples provided above but only by the claims below.